COMPUTER

http://www.phei.com.cn

普通高等教育“十一五”国家级规划教材

高等学校计算机基础及应用教材

数据结构与算法（C++语言版）

肖南峰　赵 洁　等编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

普通高等教育“十一五”国家级规划教材
华南理工大学精品课程教材
高等学校计算机基础及应用教材

数据结构与算法

（C++语言版）

肖南峰　赵　洁　等编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书为普通高等教育“十一五”国家级规划教材。

全书共分 15 章，主要内容包括：绪论、线性表、栈和队列、串、多维数组和广义表、树和二叉树、图、查找、内部排序、文件组织和外排序、贪婪算法、分而治之算法、动态规划、回溯、分枝定界法。在前 10 章中，对相应的数据结构的 ADT 描述、存储结构、基本操作、综合算法做了全面深入的阐述，每章的最后都对该章的基本内容、学习要点、具体要求、重点和难点进行了归纳和总结。在第 11～15 章中，列举了几个应用多种数据结构进行综合性算法设计的典型例子。另外，作者在参考了近年来许多的国内外教材之后，选编了大量精心设计的习题。本书每章的学习内容翔实，算法和例题典型，而且给出了对应的 VC++ 6.0 源程序。本书免费提供电子课件。

本书不仅可作为计算机学科各专业学生的教材，也适合作为广大工程技术人员和自学考试人员的参考书。

图书在版编目（CIP）数据

数据结构与算法：C++语言版/肖南峰，赵洁等编. —北京：电子工业出版社，2009.5
高等学校计算机基础及应用教材
ISBN 978-7-121-08301-3

Ⅰ.数… Ⅱ.①肖…②赵… Ⅲ.①数据结构—高等学校—教材②算法分析—高等学校—教材③C 语言—程序设计—高等学校—教材 Ⅳ.TP311.12 TP312

中国版本图书馆 CIP 数据核字（2009）第 021494 号

责任编辑：冉 哲
印 刷：
装 订：北京市李史山胶印厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×1092 1/16 印张：19.75 字数：488 千字
印 次：2009 年 5 月第 1 次印刷
印 数：4 000 册 定价：29.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

“数据结构”是计算机学科各个专业的一门重要的专业基础课程。本课程主要讲授数据的逻辑结构、存储结构、基本运算、运算实现、算法设计、算法分析、算法评价等方面的内容，使学生对线性表、栈、队列、串、数组、树及二叉树、无向图和有向图、静态及动态查找表、文件等各种数据结构有深刻的理解，对各种常见的排序方法与算法有深入的了解。在此基础上，还要求学生系统地掌握在不同的存储结构上利用上述数据结构进行综合性算法设计的方法和技巧。因此，它是一门理论性和实践性都很强的课程。

根据作者多年的教学实践发现，学生对于数据结构的应用，特别是在做算法设计习题和编程上机实习这两个环节上都不同程度地存在着一定的困难。为了帮助学生更好地掌握该课程的知识，提高算法设计和动手编程的能力，急需为计算机学科各专业开设的“数据结构”课程编写一本基础扎实、知识面广、适应性强的教材。为此，在华南理工大学精品课程建设基金的资助下，我们编写了这本《数据结构与算法（C++语言版）》教材，主要目的就是加强基础、拓宽知识面、增强适应性，以便使学生能够更深入地理解教材内容，开拓思想，培养并掌握良好的算法设计与程序实现的技能，以及解决实际问题的能力。

本书为普通高等教育“十一五”国家级规划教材。

本书共分 15 章，主要内容包括：绪论、线性表、栈和队列、串、多维数组和广义表、树和二叉树、图、查找、内部排序、文件组织和外排序、贪婪算法、分而治之算法、动态规划、回溯、分枝定界法。在前 10 章中，对相应的数据结构的 ADT 描述、存储结构、基本操作、综合算法做了全面深入的阐述，每章的最后都对该章的基本内容、学习要点、具体要求、重点难点进行归纳和总结。在第 11~15 章中，列举了几个应用多种数据结构进行综合性算法设计的典型例子。另外，作者在参考了近年来许多的国内外教材之后，选编了大量精心设计的习题。本书每章的学习内容翔实，算法和例题典型，而且给出了对应的 VC++ 6.0 源程序。本书提供免费电子课件。

本书不仅可作为计算机学科各专业学生的教材，也适合作为广大工程技术人员和自学考试人员的参考书。

肖南峰教授、黄敏讲师和张芩讲师编写了第 1～10 章并选编了全部习题，赵洁讲师编写了第 11～15 章及所有的 Visual C++ 6.0 源程序，吕建明讲师校对了部分章节的习题。在本教材的编写过程中，华南理工大学“数据结构”精品课程课题组和“智能计算机科研团队”的多位教师提出了许多的宝贵意见，我们在此向他们表示衷心的感谢。另外，还要感谢华南理工大学精品课程建设基金的支持。由于作者水平有限，教材中难免会存在错误，因此热忱地欢迎广大读者提出批评和意见。

编　者

2009 年 3 月

于华南理工大学

作 者 简 介

肖南峰博士，男，1962 年 11 月生，华南理工大学计算机科学与工程学院教授，博士生导师。1982 年 7 月毕业于华中工学院（现为华中科技大学）自动控制与计算机工程系，获工学学士学位；1989 年 1 月毕业于东北工学院（现为东北大学），获工学硕士学位；2001 年 6 月毕业于日本横浜国立大学，获工学博士学位。2001 年 9 月至 2002 年 9 月在澳大利亚 Deakin 大学从事科学研究。他作为主持人先后完成了 2 项国家自然科学基金项目、2 项广东省自然科学基金重点项目，1 项教育部留学回国人员科研启动基金项目，以及由广东省教育厅和华南理工大学等资助的 20 多项教学与科研课题，在国内外发表学术论文 120 多篇，其中被三大索引收录近 50 篇，出版专著和教材 5 部，申请或获得发明及实用新型专利 5 项，软件版权 10 项。他一直在教学一线从事“数据结构”等课程的教学，已先后为近 20 届计算机专业、计算机辅修专业、电类联合班、继续教育学院和网络学院的本科生讲授过“数据结构”、“高级程序设计语言”等课程，积累了丰富的教学经验。

目　录

第1章 绪 论

从世界上第一台计算机诞生至今，已有 60 多年的历史。在这期间，计算机的发展和应用已经渗透到了人类社会的各个领域，计算机加工和处理的对象也从纯粹的数值发展到了字符、图像、声音等各种具有一定结构的数据。为了更好地设计程序，以提高计算机在解决复杂问题时的处理效率，研究数据的特性和数据之间存在的关系至关重要。“数据结构”作为计算机科学与技术领域中的一门专业基础课，它专门研究数据的特性和数据之间存在的关系，以及如何在计算机中有效地存取数据和处理数据。因此，“数据结构”是设计和实现编译程序、操作系统、数据库系统和大型应用程序的重要基础，它也是介于数学、计算机硬件和计算机软件之间的一门核心课程，并将随着人类社会的各个领域中计算问题的不断深入研究而继续发展。

1.1 什么是数据结构

1.1.1 基本概念

（1）数据：信息的载体，是客观事物的符号表示。数据能够被计算机识别、存取和处理，数据也是计算机程序加工和处理的“原料”。例如，实数、字符串、图像和声音等。

（2）数据项：具有独立含义的最小标识单位。例如，字段、域、属性等。

（3）数据元素：数据的基本单位。一个数据元素可由若干个数据项组成。

（4）数据对象：性质相同的数据元素的集合，是数据的一个子集。例如，26 个英文字母构成的字符集合，一个学校全体学生或教师构成的学生集合或教师集合等。

（5）数据结构：相互之间存在一种或多种特定关系的数据元素的集合，即数据的组织形式。数据结构的形式化定义通常用一个二元组 Data_Structure=(D, R)来表示，式中，D 是数据元素的有限集（也即数据对象），R 是 D 上关系的有限集。

1.1.2 数据结构的内涵

数据结构一般包含数据的逻辑结构和存储结构及数据运算。数据结构的研究内容包括两方面：一方面是抽象地研究数据集合的结构（抽象数据结构）特点；另一方面是研究如何把抽象数据结构转化为存储组织形式（数据的存储结构）。这两方面的研究既可独立于各个领域的应用来研究，也可结合具体应用领域中的特点来进行研究。例如，专门研究计算机图像识别或定理证明中的数据结构。目前，从抽象数据类型的观点来讨论数据结构已经成为一种新的趋势，并越来越被人们所重视。

1. 数据的逻辑结构

数据的逻辑结构是指数据元素以及它们相互之间的逻辑关系，数据的逻辑结构与数据的存储无关。根据数据元素之间关系的不同特性，通常有 4 类逻辑结构：① 集合，集合的

逻辑结构中所有数据元素都属于同一个集合，所有数据元素杂乱无章地聚集在一起，各个数据元素之间无任何联系；② 线性结构，逻辑结构中的数据元素之间存在着一个对一个的关系，各个数据元素之间通常有严格的先后次序关系；③ 树形结构，逻辑结构中的数据元素之间存在着一个对多个的关系，各个数据元素之间通常有严格的层次关系；④ 图状结构，逻辑结构中的数据元素之间存在着多个对多个的关系，各个数据元素之间均可能存在相互联系。

在不产生混淆的前提下，常将数据的逻辑结构简称为数据结构。除了上述 4 类逻辑结构之外，根据数据元素（结点）之间的前后相邻关系，数据的逻辑结构还可分为线性结构和非线性结构两大类：① 线性结构的逻辑特征是，若结构是非空集，则有且仅有一个开始结点和一个终端结点，并所有结点都最多只有一个直接前驱结点和一个直接后继结点。线性表是一个典型的线性结构，栈、队列和串等都是线性结构；② 非线性结构的逻辑特征是，一个结点可能有多个直接前驱和直接后继。树和图都是非线性结构。

【例 1-1】 怎样描述数据的逻辑结构？

对数据元素之间关系的描述是数据的逻辑结构，它可形式地用一个二元组表示为 $K=(D, R)$，式中，D 是由有限个结点所构成的集合，R 是由有限个关系所构成的集合。有时为了直观起见，也用以图示法来表示数据的逻辑结构。逻辑结构与使用的计算机无关。例如，一批数据的逻辑结构 $K=(D, R)$，式中，$D=\{d_1, d_2, \cdots, d_9\}$，$R=\{<d_1, d_2>,<d_1, d_3>,<d_1, d_4>,<d_1, d_7>,<d_1, d_8>,<d_4, d_5>,<d_4, d_6>,<d_8, d_9>\}$，则该批数据的逻辑结构如图 1.1 所示。对于 R 中包含有多种关系的情况，也可用类似的方法描述。

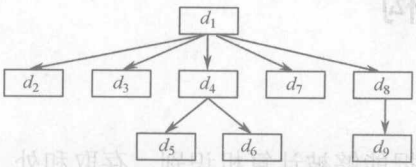

图 1.1 数据的逻辑结构示例

2. 数据的存储结构

数据的存储结构（物理结构）是指数据在计算机中的存储表示，它包括数据元素的表示和关系的表示。数据的存储结构有以下 4 种基本存储方法。

① 顺序存储。该存储方法把逻辑上相邻的结点存储在物理位置上相邻的存储单元中，结点间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构（sequential storage structure）。该方法通常借助于高级程序语言中的数组来描述，且主要应用于线性的数据结构。非线性的数据结构也可通过线性化的方法来实现顺序存储。

② 链接存储。该方法不要求逻辑上相邻的结点在物理位置上也相邻，结点之间的逻辑关系由附加的指针字段表示。由此得到的存储表示称为链式存储结构（linked storage structure），通常借助于高级程序语言的指针类型来描述。

③ 索引存储。该方法通常在存储结点信息的同时，还要建立附加的索引表。索引表由若干索引项组成。若每个结点在索引表中都有一个索引项，则该索引表称之为稠密索引（dense index）。若一组结点在索引表中只对应一个索引项，则该索引表称为稀疏索引（spare index）。索引项的形式一般是（关键字，地址），关键字是能唯一标识一个结点的那些数据项。稠密索引中索引项的地址指示结点所在的存储位置，稀疏索引中索引项的地址指示一组结点的起始存储位置。

④ 散列存储。该方法根据结点的关键字直接计算出该结点的存储地址。

以上述 4 种基本存储方法既可单独使用，也可组合起来对数据结构进行存储映射。同一种逻辑结构采用不同的存储方法，可得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构，要视具体的应用要求而定，主要考虑运算方便和算法的时空效率要求。

3. 数据的运算

数据的运算是对数据施加的操作。数据的运算定义在数据的逻辑结构上，每种逻辑结构都有一个运算的集合。在数据结构中，运算不仅仅是加、减、乘、除等运算，大多数的运算都涉及算法的实现问题，算法的实现与数据的存储结构是密切相关的。

1.1.3 数据类型和抽象数据类型

数据类型是一个值的集合和定义在这个值的集合上的一组操作的总称，通常它可看作是高级程序设计语言中已经实现的数据结构。例如，C 语言中的整型变量，其值的集合为某个区间上的整数（区间大小依赖于不同的机器），定义在其上的操作为加、减、乘、除和取模等算术运算。按“值”的不同特性，在高级程序语言中可分为：① 原子类型，其值不可分解，例如，C 语言中的基本类型（整型、实型、字符型和枚举类型）、指针类型和空类型；② 结构类型，其值是由若干个成分按某种结构组成的，故可分解，其成分可以是非结构的，也可以是结构的，例如，数组的值由若干分量组成，每个分量可能是整数，或者是数组。

抽象数据类型（abstract data type，ADT）是指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，也即，不论其内部结构如何变化，只要它的数学特性不变，都不会影响其外部的使用。抽象数据类型可表示为一个三元组（D, R, P），式中，D 是数据对象，R 是 D 上的关系集，P 是对 D 的基本操作集。本书采用以下格式定义抽象数据类型：

```
ADT 抽象数据类型名{   数据集合：<数据对象的定义>
                      数据关系：<数据关系的定义>
                      数据操作：<数据操作的定义>
                  } ADT 抽象数据类型名
```

其中，数据对象和数据关系的定义用伪码描述，基本操作的定义格式为：

```
数据操作名(参数表)
输入：<输入条件描述>
输出：<输出结果描述>
```

【例 1-2】 抽象数据类型（ADT）的定义、特点、实现与数据结构的区别是什么？

ADT 是高级程序设计语言中数据类型概念的推广，它是一个数学模型和定义在该模型上操作集合的总称。ADT 的实现方法是，将 ADT 转换成现有高级程序设计语言的说明语句，加上对应于该 ADT 中每个操作的过程或函数，也即，用现有高级程序设计语言能够支持的适当数据结构来表示 ADT 中的数学模型，并用一组过程或函数来实现定义在该模型上的各个操作。数据结构则是利用该语言的基本数据类型和复合数据的构造机制来构成的。例如，数组和记录就是 C++语言中两种主要的复合数据的构造机制。根据 ADT 定义，如果在

相同的数学模型上定义两个不同的操作集，则认为它们代表两个不同的抽象数据类型。故相同的数学模型以及在其上所定义的操作有可能在不同的 ADT 中出现。

1.2 算法和算法分析

1.2.1 算法的描述

算法（algorithm）定义：为了解决某一类问题而设计的一个有限长的操作序列。一个算法必须满足以下 5 个重要特性。

① 有穷性。算法对于任意合法的输入值，在执行有限步之后一定能结束。

② 确定性。算法中的每一个操作必须有确切的含义，无二义性，并在任何条件下，算法都只有一条执行路径。

③ 可行性。算法中的所有操作都可通过已经实现的基本运算有限次地实现。

④ 输入。算法具有零个或多个输入，这些输入为一组特定的数据对象集合。

⑤ 输出。算法具有一个或多个输出，它是一组与“输入”有确定关系的量值。

1.2.2 算法设计的要求

（1）正确性（correctness）

算法的执行结果应当满足预先规定的 4 个要求：①程序不含语法错误；② 程序对于几组输入数据能够得出满足规格说明要求的结果；③程序对于精心选择的典型、苛刻且带有刁难性的几组输入数据能够得出满足规格说明要求的结果；④程序对于一切合法的输入数据都能产生满足规格说明要求的结果。

（2）可读性（readability）

算法应有助于人们阅读、理解和调试，晦涩难懂的算法易于隐藏较多错误，难以调试和修改。

（3）健壮性（robustness）

当输入不合法的数据时，算法能够做出适当的反应或处理，不至于产生莫名其妙的结果。同时，处理出错的方法应该是返回一个表示错误或错误性质的值，并终止程序的执行，以便在更高的抽象层次上进行处理。

（4）时空效率（efficiency）

要求算法执行的时间应该尽可能短、算法执行过程中占用的存储空间应该尽可能少。时空要求与求解问题的规模有关，两者通常相互矛盾，因此，应在它们之间有所平衡。

1.2.3 算法分析

算法分析的两个主要方面是分析算法的时间复杂度和空间复杂度，主要考察算法的时间效率和空间效率，以便比较和改进算法。通常，在算法的运算空间较为充裕的情况下，更多地关注算法的时间复杂度。

1. 时间复杂度

算法执行的时间可通过依据该算法编制的程序在计算机上从开始运行到结束运行所消

耗的时间来度量，也就是算法中每条语句的执行时间之和。由于每条语句的执行时间是该语句重复执行的次数或频度（frequency count）与该语句执行时间的乘积，而语句的执行时间又与机器性能、编译程序等诸多因素有关，难以统一和确定。因此，假设每条语句的执行时间为一个单位时间，则算法的执行时间为算法中所有语句的频度之和。

一般而言，算法中基本操作的频度是问题规模 n（如算法所处理的矩阵的阶数，线性表的长度）的某个函数 $f(n)$，算法的时间量度记作 $T(n)=O(f(n))$，它表示随问题规模 n 的增大，算法的执行时间增长率与 $f(n)$的增长率相同，称为算法的渐进时间复杂度（asymptotic time complexity），简称时间复杂度。同时，要全面地分析算法，需要分别考虑算法在最坏情况、最好情况以及平均情况下的时间代价。对于最坏情况下的时间复杂度，主要采用大写数学符号 O 表示法来描述。一般定义为：当且仅当存在正整数 c 和 n_0，使得 $T(n)\leqslant c\ f(n)$对所有的 $n\geqslant n_0$ 成立，则称该算法的渐进时间复杂度为 $T(n)=O(f(n))$。

在一般情况下，对于一个问题（算法）只需选择一种基本操作来讨论算法的时间复杂度即可，有时也需要同时考虑几种基本操作，甚至可对不同的操作赋以不同权值，以反映执行不同操作所需的相对时间，这种做法便于综合比较解决同一问题的两种完全不同的算法。由于算法的时间复杂度考虑的只是对于问题规模 n 的增长率，因此在难以精确计算基本操作执行次数（或语句频度）的情况下，只需求出它关于 n 的增长率或阶即可。

【例 1-3】 如何进行算法的时间复杂度分析。

首先介绍计算增长率的加法规则和乘法规则。设 $T_1(n)$和 $T_2(n)$分别是程序段 P_1 和 P_2 的运行时间，且 $T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，即 $T_1(n)$是 $f(n)$的函数，$T_2(n)$是 $g(n)$的函数（O 函数定义见后），则执行 P_1 之后紧接着执行 P_2 的运行时间为：$T_1(n)+T_2(n)=O(\max\{f(n),g(n)\})$，称为加法规则；$T_1(n)\times T_2(n)=O[f(n)\times g(n)]$，称为乘积规则。一般来说，分析程序的时间复杂度是，先求出各模块（各语句）的运行时间，再求整个程序的运行时间，它可表示成唯一参数——输入数据的规模 n 的函数。具体可遵循以下规则。

① 每个赋值或读/写语句的运行时间通常是 $O(1)$。如果赋值语句的右部为函数调用，则要考虑计算函数值所消耗的时间。

② 序列语句的运行时间由加法规则确定。

③ 语句 if B then S_1 else S_2 的运行时间为条件 B 的测试时间（通常取 $O(1)$）加上两个分支语句 S_1、S_2 运行时间的较大者。若无 else S_2，则只需加上 S_1 的运行时间。

④ 循环语句的运行时间是循环体本身的运行时间和计算循环参数、测试循环终止条件及跳回循环开头所花的时间，后一部分通常取 $O(1)$。遇到多层循环时，要由内层向外层逐层分析。在分析外层循环时间时，内层循环的运行时间应该是已知的，可把内循环看成是外循环体的一部分。

⑤ 若程序中只包含非递归过程，则从没有调用语句（对函数也认为是调用）的过程开始，计算所有这种过程的运行时间。然后考虑有调用语句的任一过程 P，在 P 调用的全部过程的运行时间都算完之后，即可开始计算 P 的运行时间。若在程序中有递归过程，则可令每个递归过程对应于一个未知的时间开销函数 $T(n)$，其中 n 是过程参数的长度。之后，列出一个关于 T 的递归方程并求解之。

【例 1-4】 下面是一个 $n\times n$ 阶矩阵 A 自乘得到 $B=A\times A$ 的算法，分析其时间复杂度。

```
int  A[n][n];                                                    //全局数组
void  mtxmlt()
{       int B[n][n];                                             //语句频度
        for (int i=0; i<n; i++)                                  //n+1
                for(int j=0; j<n; j++)                           //n×(n+1)
                {   B[i][j]=0;                                   //n×n
                    for(int k=0; k<n; k++)                       //n²×(n+1)
                        B[i][j]=B[i][j]+A[i][k]*A[k][j];         //n×n×n
                }
}
```

时间复杂度 $T(n)=n+1+n(n+1)+n\times n+n^2(n+1)+n\times n\times n=2n^3+3n^2+2n+1$。当 $n\to\infty$ 时，$T(n)\propto n^3$，故算法时间复杂度的数量级为 $O(n^3)$。

【例 1-5】 已知算法如下，求带下划线语句的频度。

```
int i=0;
    while((i<n)&&(x!=A[i])) i++;
        if(A[i]==x)  return i;
```

在此程序段中，语句的频度不仅是 n 的函数，而且与 x 及数组 A 中各分量的值有关。在这种情况下，通常考虑最坏的情况。由于 while 循环执行的最大数为 $n-1$，因此下划线语句频度为 $n-1$。

【例 1-6】 分析计算 $n!$ 的递归函数 fact(n)的时间复杂度。

```
long  fact(int n)
{   if (n==1)  return 1;
    return  fact(n-1)*n;
}
```

递归函数 fact(n)的输入规模是 n，设 $T(n)$是 fact(n)的时间开销函数。在上述算法中，if 语句条件测试及语句 return 的运行时间是 $O(1)$，递归调用 fact($n-1$)的运行时间是 $T(n-1)$。假设两个整数相乘和赋值操作的运算时间是 $O(1)$，故 return fact($n-1$)*n 的运行时间是 $O(1)+T(n-1)$。因此，对于常数 C 和 D 有 $T(n)=D$，$n\leqslant 1$；$T(n)=C+T(n-1)$，$n>1$。现在来解这个递归方程。设 $n>2$，对上式中 $T(n-1)$进行展开有 $T(n-1)=C+T(n-2)$，代入 $T(n)$中，有 $T(n)=2C+T(n-2)$，再展开 $T(n-2)$得 $T(n)=3C+T(n-3)$。一般有 $T(n)=iC+T(n-i)$，$n>i$。最后，当 $i=n-1$ 时，得 $T(n)=C(n-1)+T(1)=C(n-1)+D$。当 $n\to\infty$ 时，$T(n)\propto n$。

2. 空间复杂度

算法在执行时需要占用一定的存储空间，这些空间除了包括程序、输入数据、常数、变量所占的空间外，还包括算法对输入数据进行运算以及为实现运算所需信息的额外空间。额外空间与算法的质量密切相关，好的算法既节省时间又节省额外空间。如果算法的输入数据所占用的空间只取决于问题本身，与算法无关，则算法的存储空间只需要分析除输入数据和程序之外的额外空间；否则，应同时考虑输入数据本身所需空间（与输入数据的表示形式